

A new approach to joint resource management in MEC-IoT based federated meta-learning

Faustin Samafou^{1,3}, Bakhit Amine Adoum^{1,2,4}, Ado Adamou Abba Ari^{3,5,7}, Faïtchou Marius Fidel¹, Amir MOUNGACHE⁴, Nasrullah Armi⁶, Abdelhak Mourad Gueroui⁷

¹National Higher School of Information and Communication Technologies (ENASTIC), N'Djamena, Chad

²Electronics, Electrotechnics, and Energy Research Laboratory (LR3E), Department of Electrical Engineering, National Higher Institute of Science and Technology of Abeche, Abeche, Chad

³LaRI Lab, University of Maroua, Maroua, Cameroon

⁴Laboratory of Information and Communication Technologies (LARTIC), Faculty of Exact and Applied Sciences of Farcha, University of N'djamena, Ndjamen, Chad

⁵CREATIVE, Institute of Fine Arts and Innovation, University of Garoua, Garoua, Cameroon

⁶National Research and Innovation Agency, Research Center for Telecommunication, Bandung, Indonesia

⁷DAVID Lab, Université Paris-Saclay, University of Versailles Saint-Quentin-en-Yvelines, Versailles, France

Article Info

Article history:

Received Dec 6, 2023

Revised Feb 15, 2024

Accepted Mar 20, 2024

Keywords:

Federated learning

Meta-learning

Mobile edge computing

Resource allocation

Task offloading

ABSTRACT

MEC and IoT are rapidly expanding technologies that offer numerous opportunities to enhance efficiency and application performance. However, the huge volume of data generated by IoT devices, coupled with computational and latency constraints, poses data processing challenges. To address this within the MEC architecture, deploying computing servers at the network edge near IoT devices is a promising approach. This reduces latency and traffic load on the core network while improving the user experience. However, offloading computations task from IoT devices to MEC servers and efficiently allocating computing resources is a complex problem. IoT tasks may have specific requirements in terms of latency, bandwidth and energy efficiency, while computing resources and capacities maybe limited or shared between several users. We propose an approach called FedMeta2Ag, which we evaluate using the MNIST database. With 20 epochs, the training accuracy reached 91.5%, while the test accuracy achieved 92.0%. Performance consistently improved during the initial 20 iterations and gradually stabilized thereafter. Additionally, we compared the performance of our proposed model with existing methods, finding that our approach outperforms existing models in predicting performance more accurately. Thus, this approach effectively meets the demanding performance requirements of wireless communication systems.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Bakhit Amine Adoum

National Higher School of Information and Communication Technologies (ENASTIC)

P.O. Box 5363 N'Djamena, Chad

Email: bakhitamine@yahoo.fr

1. INTRODUCTION

Mobile edge computing (MEC) and internet of things (IoT) are transformative technologies that have revolutionized the way data is processed and applications are delivered [1], [2]. MEC brings computation and storage capabilities closer to the network edge, enabling real-time data processing and reducing latency [3]. IoT,

on the other hand, connects a multitude of smart devices and sensors, allowing for seamless communication and data exchange [4]. The convergence of MEC and IoT offers significant opportunities for innovative applications and services [5]. However, effective resource management is critical to fully exploit the potential of these technologies.

Resource management involves allocating computing resources, such as computational power, memory, and network bandwidth, in a manner that optimizes performance and ensures efficient utilization [6]. FedMeta2Ag has been identified as a new approach to address the challenges of resource management in MEC-IoT environments [7]. Federated approach enables collaborative learning on decentralized data while preserving privacy and security [8]. It allows multiple IoT devices to train a shared model without the need to transfer sensitive data to a central server. MetaL, on the other hand, leverages past experiences to enable models to learn and adapt quickly to new tasks or environments [9].

However, resource management in MEC-IoT based FedMeta2Ag faces several challenges [10]. First, the dynamic and heterogeneous nature of MEC-IoT environments necessitates adaptive resource allocation (ARA) strategies that can efficiently utilize available resources. IoT devices vary in terms of computational capabilities, network connectivity, and energy constraints, making it challenging to allocate resources effectively. Additionally, the resource demands of different applications or tasks may vary over time, requiring the system to be capable of dynamically adjusting resource allocation [11]. Furthermore, privacy and security are paramount concerns in federated learning (FL). The distributed nature of the data and the need to protect sensitive information pose challenges in ensuring privacy-preserving operations and secure model aggregation. Balancing privacy requirements with the need for efficient resource utilization adds an additional layer of complexity to resource management in FedMeta2Ag [12].

In the context of resource management in MEC-IoT environments, several research studies have addressed different aspects of the problem. Moghaddasi and Rajabi [13] proposed a double deep Q-learning approach for energy-efficient resource management in device-to-device (D2D) MEC environments. Their work focused on optimizing energy consumption but did not consider the privacy-preserving and security aspects of FL in resource management. In [14], [15] proposed a joint computation offloading and resource allocation scheme for MEC-IoT systems to optimize latency and energy consumption. While their approach addressed the optimization objectives, it did not incorporate FL or MetaL techniques in the resource management process. Dandoush *et al.* [16], addressed the privacy concerns in FL-based resource management by proposing a privacy-preserving approach. However, their work did not consider the integration of MetaL techniques for ARA in MEC-IoT frameworks.

In a more recent study, proposed a reinforcement learning-based approach for dynamic resource allocation (DRA) in MEC-IoT systems [17]. Although their work addressed DRA, it did not explore the potential of FL or MetaL techniques in resource management. Furthermore, introduced a hierarchical resource management (HRM) framework using deep reinforcement learning (DRL) in MEC-IoT networks [18]. While their approach provided a hierarchical structure for resource allocation, it did not specifically focus on the integration of FL or MetaL techniques.

A similar study proposes DRL techniques to efficiently handle task offloading and resource allocation in MECs [19]. The authors have developed DRL methods for solving optimization problems in MECs, focusing on task offloading and service migration. These methods enable systems to autonomously learn optimal solutions to the two optimization problems mentioned above. To further improve the efficiency of the DRL-based task offloading method, the authors have integrated reinforcement MetaL. In addition, for the service migration problem, they exploited the DRL-based solution. The results are satisfactory compared with the heuristic algorithms.

Li *et al.* [20] conducted research on the task offloading problem in MEC systems and proposed a distributed algorithm to address the task scheduling problem for end devices. Their approach leverages spatial correlation features in the scenario and efficiently schedules multiple tasks. They also investigated the collaborative computing scheduling problem in terms of processing time and bandwidth usage. The objectives were to minimize long-term energy consumption and schedule tasks efficiently based on latency requirements. However, a limitation of their work is that the algorithm needs to be retrained for new scenarios and requires devices to broadcast their relevant information during the training process.

Al-Ameer and Bhaya [21] have proposed a FL approach to resource management in MEC-IoT systems. However, MetaL techniques were not integrated into the federated approach they proposed for ARA. Recently, Adoum *et al.* [22] proposed a resource allocation approach based on a FedMeta2Ag framework for

MEC-IoT environments.

Given the existing research gaps and limitations, there is a need for a comprehensive approach that integrates FL and MetaL techniques to address the resource management challenges in MEC-IoT environments. Such an approach should consider energy efficiency, privacy preservation, security, adaptability to dynamic conditions, and efficient allocation of computing resources. By incorporating both FL and MetaL, it would be possible to achieve efficient resource allocation while preserving privacy, optimizing performance, and enhancing the capabilities of MEC-IoT systems.

In this article, we propose a new approach to joint resource management in MEC-IoT based FedMeta2Ag that tackles these challenges. By integrating FL and MetaL techniques, we aim to develop a framework that not only optimizes resource allocation but also ensures privacy preservation and security in MEC-IoT environments. Our research aims to provide a comprehensive solution that addresses the resource management challenges posed by dynamic and heterogeneous MEC-IoT systems, enabling efficient utilization of computing resources and enhancing the performance of applications and services. Our main contributions are summarized:

- Proposal of a method for task offloading and resource allocation in the context of MEC-IoT using a meta-FL approach. The aim is to optimize task and resource allocation in MEC environments for IoT applications.
- Proposal of a new federated method that enables MetaL in MEC environments. The proposed approach enables collaborative learning of multiple edge IoT devices and enhancement of their task offloading capabilities while preserving data confidentiality and reducing communication overheads.
- Design of a MEC-IoT framework based on double aggregation Meta-FL. Our framework uses a double aggregation approach in the FL process adapted to MetaL in the MEC-IoT. The proposed method aims to improve the aggregation step in the FL process, enhancing knowledge sharing and model convergence between edge devices.
- Development of an algorithm called FedMeta2Ag and its extension integrating reinforcement learning (FedMetaR2Ag).

The remainder of the article is organized as: section 2 presents the system model and problem formulation based on FedMeta2Ag for an MEC-IoT environment. In this section, the system architecture, communication model, local task execution, and task offloading to the MEC server are presented and described. The problem of joint resource management based on FL in the MEC-IoT context is also formulated. Furthermore, our algorithm, FedMeta2Ag, and its extension using reinforcement learning techniques adapted to dynamic environments are presented and described. In section 3, through repeated simulations, we evaluated our FedMeta2Ag algorithm considering accuracy rate and model loss as performance criteria, across different learning epochs during training and testing phases. Finally, we discuss and compare the results of our proposed approach with the results of other similar works. Section 4 summarizes the main contributions of our approach and presents some perspectives.

2. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we present the system architecture, communication model, local task execution and task downloaded to the local MEC servers (MEC-L) for our proposed joint resource management approach in MEC-IoT based FedMeta2Ag. The architecture is composed of three tiers: the IoT user devices, the MEC-L server, and the central server. Each IoT user device includes a local compute unit and a wireless communication module. The MEC-L server provides local computing resources and storage, while the central server handles the global parameter updates and model aggregation.

2.1. System architecture

In this subsection, we present a federated method for MetaL within the MEC framework. The training system is organized with a decentralized structure, similar to the traditional federated approach. We incorporate the model-agnostic meta-learning (MAML) strategy into the federated framework and carry out a dual model aggregation using MEC-based model aggregation to achieve improved model performance and task offloading simultaneously. After the global model is thoroughly trained, individual testing is conducted following a few gradient descent fine-tuning steps.

The training system is configured with a decentralized structure, identical to the conventional FL approach. We implement the MAML strategy in the federated framework and perform a dual model aggregation

based on MEC based model aggregation to simultaneously achieve better model performance and better task offloading. Once the global model is well trained, the test is carried out individually after a few gradient descent fine-tuning steps.

We consider a system with MEC-L server placed at the user's periphery and a central MEC server (MEC-C) as illustrated in Figure 1. The set of MEC-L server and MEC-C is denoted $\mathcal{M} = \{1, 2, \dots, M\}$. There are N IoT devices (ID) noted $\mathcal{N} = \{1, 2, \dots, N\}$. Each IoT device has a task that is assumed to be atomic (cannot be divided). Tasks are executed locally, or downloaded to the MEC-L server for processing.

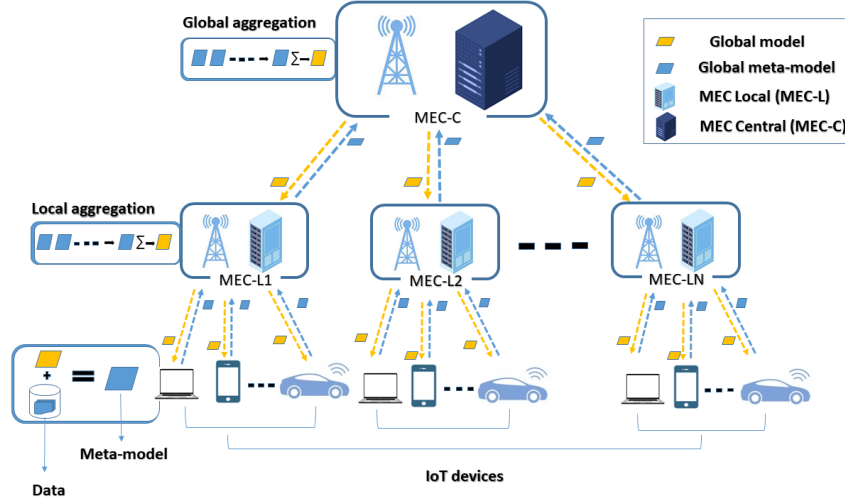


Figure 1. Model for task offloading and resource allocation based on FedMeta2Ag in MEC

The unloaded task must be processed locally by the central processing unit (CPU). The task requiring unloading must be transmitted to the MEC-L server and processed. To avoid bottleneck at the MEC-L server level, each server is only responsible for a certain number of IoT end-devices, where a generalized model is obtained after aggregation to enable new users to use the model directly, thus reducing the learning time. Each MEC-L server obtains its own global model. The process is iterative, until a better model is found.

For more collaborative and to have the same high-performance model for all devices, the general model obtained is then transferred to the MEC-C server to obtain a more general model after global aggregation. This more generalized model is then sent to all the MEC-L edge servers, which in turn dispatch it to the IoT devices.

The computational task on each IoT device T_i is defined as (F_i, D_i, T_i^{max}) for any i element in the set \mathcal{N} [23]. Here, F_i represents the total number of CPU cycles, D_i indicates the size of the data transmitted from the IoT devices to the MEC-L during the task offload phase, and T_i^{max} is the maximum tolerated latency. The values of D_i , F_i and T_i^{max} are determined using the methods described in [24], [25]. The binary task unloading strategy is considered as:

$$X = \{x_{ij} \in \{0, 1\} | i \in \mathcal{N}, j \in \mathcal{M}\} \quad (1)$$

$x_{ij} = 1$ indicates that the i^{th} IoT device decides to offload the task to the j^{th} server MEC-L, while $x_{ij} = 0$ indicates that the i^{th} IoT device decides not to unload the task on the j^{th} MEC-L.

2.1.1. Communication model

Our proposed approach employs a communication model where each user device communicates with the MEC-L server using a get-compute-set protocol. When a device has a learning task to perform, it sends a request to the MEC-L server to obtain computational resources. The MEC-L server then allocates the necessary compute resources and downloads the model parameters locally. Once the resources are allocated, the device performs the local computations and sends the results back to the MEC-L server for storage and aggregation. The device delegates the computation task to the MEC server, the uplink and downlink are represented as Rayleigh channels, with the transmission rate of each device being defined according to the principles described in [26]:

$$r_{ij} = B \log_2 \left(1 + \frac{P_{ij} h_{ij}}{\sigma^2} \right), \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \quad (2)$$

where B is the bandwidth, P_{ij} is the transmission power from the i^{th} IoT device to the j^{th} MEC-L, σ^2 denotes the noise power and h_{ij} is the channel gain which is expressed as:

$$h_{ij} = \frac{\beta_0 l_{ij}}{(X_j - x_i)^2 + (Y_j - y_i)^2}, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \quad (3)$$

where β_0 is the channel power gain, l_{ij} is the fading factor, $(X_j - Y_j)$ the coordinates of the j^{th} MEC, $(x_i - y_i)$ the coordinates of the i^{th} IoT device. Consequently, considering the throughput expressed above, the time and energy consumed are defined by [27]. The time taken to offload data from the i^{th} IoT device to the j^{th} MEC-L is defined by:

$$T_{ij} = \frac{D_i}{r_{ij}}, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \quad (4)$$

The energy consumed is expressed as:

$$e_i = T_{ij} p_{ij}, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \quad (5)$$

2.1.2. Local task execution

When a user device receives a task, it first checks if it has sufficient local resources to perform the computation. If it does, it performs the computations locally and sends the results back to the MEC-L server. If the device's local resources are insufficient, it sends a request to the MEC-L server for additional resources. The MEC-L server then evaluates if it has enough resources to allocate before sending the results back to the device.

If the IoT device decides to process the task locally, the additional cost has two components: local processing latency and local power consumption, since no task offload is involved. These two components are determined in accordance to formulas taken from [28]. The local computation time is:

$$T_i^l = \frac{F_i}{f_i^l}, \quad \forall i \in \mathcal{N} \quad (6)$$

where f_i^l denotes the computing power of the i^{th} IoT device's CPU (i.e. the number of cycles per second). The local processing power consumption is:

$$e_i^l = k(f_i^l)^2 F_i, \quad \forall i \in \mathcal{N}; \quad (7)$$

where k , is the power consumption factor per CPU cycle of the IoT device and is equal to: $k = 10^{-20}$.

2.1.3. Task downloaded to the MEC-L server

When a user device requests resources from the MEC-L server, the task is downloaded to the server. The server then identifies an appropriate model to use and aggregates the results from the user devices to optimize the model. Once the model has been optimized, the server sends the updated model parameters back to the user devices to continue the learning process.

When unloading a task, data is sent to the MEC-L server. The energy calculations on the MEC server side are ignored [29], [30]. The server subsequently sends the computation and processing results back to the terminal device in a data format.

Given that the downloaded data size is notably larger than the data returned by the server, transmission delay during the return of computation results to the user device is ignored [31], [32].

When the IoT device decides to offload a computational task to the MEC-L server for processing, the server must allocate corresponding computational resources to the offloaded computational task. The task execution time of the i^{th} device on the j^{th} MEC-L is:

$$T_{ij}^{exe} = \frac{F_i}{f_{ij}^{MEC}}, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \quad (8)$$

where f_{ij}^{MEC} MEC is the capacity of the j^{th} MEC allocated to the i^{th} IoT device. When the IoT device uploads a computation task to the MEC server, the resource allocation policy is defined as:

$$F = \{f_{ij}^{MEC} \mid i \in \mathcal{N}, j \in \mathcal{M}\} \quad (9)$$

The global time consumption for a task is:

$$T_{ij}^{total} = T_{ij} + T_{ij}^{exe}, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \quad (10)$$

The energy consumed during unloading is:

$$e_{ij}^{MEC-L} = p_{ij} T_{ij}, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \quad (11)$$

2.2. Problem formulation

In this subsection, the problem is formulated based on the various equations. The time consumption for each IoT device is considered as:

$$T_i = \sum_{j \in \mathcal{M}} x_{ij} (T_{ij} + T_{ij}^{exe}) + (1 - x_{ij}) T_i^l, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \quad (12)$$

Total energy consumption is expressed as:

$$E_i = \sum_{j \in \mathcal{M}} x_{ij} e_{ij}^{MEC-L} + (1 - x_{ij}) e_i^l, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \quad (13)$$

The objective function $Q(X, F)$ is defined as:

$$Q(X, F) = \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} (\lambda_i^T T_i + \lambda_i^E E_i) \quad (14)$$

where λ_i^T and λ_i^E are weighted coefficients. The weighted coefficients are determined by specific application scenarios. $\lambda_i^T > \lambda_i^E$ if the task is urgent. If the task is energy-sensitive, $\lambda_i^E > \lambda_i^T$ can be implemented. The weighted coefficients in energy and computation time can be determined by applying multicriteria decision-making theory [33].

The objective of the mathematical model is to minimize the latency and energy consumption of all IoT devices subject to latency requirements and limited resources is as:

$$\begin{aligned} & \min_{X, F} Q(X, F) \\ & St : \\ & C_1 : T_{ij}^{trans} \leq T_i^{max} \\ & C_2 : \sum f_{ij}^{MEC} \leq F_{j,max}^{MEC-L} \\ & C_3 : 0 \leq f_{ij}^{MEC} \leq F_{j,max}^{MEC-L} \\ & C_4 : \sum f_i^l \geq 0 \\ & C_5 : \lambda_i^T + \lambda_i^E = 1 \\ & C_6 : x_{ij} \in \{0, 1\} \\ & C_7 : x_{ij} + x_{i0} = 1 \end{aligned} \quad (15)$$

where $F_{j,max}^{MEC-L}$ is the computational capacity of the j^{th} MEC-L. With regard to constraints, C_1 indicates that the transmission time must not exceed the maximum time. C_2 and C_3 indicate that the total peripheral computing resources allocated to all tasks must be the maximum computing capacity of the MEC-L server. C_4 indicates that the IoT devices have local computing capacity. C_5 indicates that the task must be either urgent, or energy-sensitive, but not both, hence: $\lambda_i^T + \lambda_i^E = 1$. C_6 and C_7 indicate that the device can make only one decision at any given time.

Observing both the presence of discrete (X) and continuous (F) variables, it is clear that that Q is a mixed-integer nonlinear programming (MINLP) problem, is non-convex and is non-convex and NP-hard [30], and that it is difficult to solve.

Therefore, the problem is decomposed into two subproblems: task unloading decision subproblem, taking into account constraints C_6 and C_7 , and constraints C_2 to C_3 for the resource allocation sub-problem.

In this article, task offloading is solved using the FedMeta2Ag algorithm and the convex optimization technique for resource allocation.

Considering the task unloading decision as a series of discrete binary variables, we design our algorithm, which can quickly learn the optimal unloading decision X^* . The (16) is written as:

$$\begin{aligned} \min_X Q_1(X) \\ \text{St } C_6, C_7 \end{aligned} \quad (16)$$

once X^* is identified, the problem can be reformulated as:

$$Q_2 = \min_F Q(X^*, F) \quad (17)$$

with regard to f_{ij}^{MEC} , the second derivative of the fraction $\lambda_i^T (\frac{F_i}{f_{ij}^{MEC}})$ gives us:

$$\begin{aligned} \frac{\partial Q(X^*, F)}{\partial f_{ij}^{MEC}} &= -\frac{\lambda_i^T F_i}{(f_{ij}^{MEC})^2}, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \\ \frac{\partial^2 Q(X^*, F)}{\partial (f_{ij}^{MEC})^2} &= 2\frac{\lambda_i^T F_i}{(f_{ij}^{MEC})^3}, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \end{aligned} \quad (18)$$

According to the convexity theorem for functions, this problem is categorized as a convex optimization problem due to the second-order derivative being greater than zero. There is then an absolute minimum at point $F = F^*$. Consequently, the problem satisfies the Slater condition, and the Karush-Kuhn-Tucker (KKT) optimality conditions can be used to find the optimal allocation of resources at a point F^* . The function Q_2 , which represents the objective, is defined as:

$$\begin{aligned} \Lambda(F, \beta) &= \min_F \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} \lambda_i^T \left[\left(\frac{D_i}{r_{ij}} + \frac{F_i}{f_{ij}^{MEC}} \right) + \frac{F_i}{f_i^{IoT}} \right] + \\ &\lambda_i^E \left[P_{ij} \frac{D_i}{r_{ij}} + k(f_i^{IoT})^2 F_i \right], \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \end{aligned}$$

Applying the Lagrange multiplier method, the Lagrangian function for $\Lambda(F, \beta)$ can be expressed as:

$$\begin{aligned} \mathcal{L}(\Lambda(F, \beta)) &= \min_F \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} \lambda_i^T \left[\left(\frac{D_i}{r_{ij}} + \frac{F_i}{f_{ij}^{MEC}} \right) + \frac{F_i}{f_i^{IoT}} \right] + \\ &\lambda_i^E \left[P_{ij} \frac{D_i}{r_{ij}} + k(f_i^{IoT})^2 F_i \right] + \beta \left(\sum_{j \in \mathcal{M}} f_{ij}^{MEC} - F_{j, \max}^{MEC-L} \right), \\ &\forall i \in \mathcal{N}, \forall j \in \mathcal{M} \end{aligned} \quad (19)$$

where β is the Lagrange multiplier associated with constraints (C_2, C_3) on computational resources satisfying $\beta \leq 0$. The partial derivative of $\mathcal{L}(\Lambda(F, \beta))$ with respect to f_{ij}^{MEC} gives us:

$$\frac{\partial \mathcal{L}(\Lambda(F, \beta))}{\partial f_{ij}^{MEC}} = -\lambda_i^T \frac{F_i}{(f_{ij}^{MEC})^2} + \beta, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \quad (20)$$

In accordance with the KKT condition:

$$\frac{\partial \mathcal{L}(\Lambda(F, \beta))}{\partial f_{ij}^{MEC}} = 0 \quad (21)$$

The optimal solution for f_{ij}^{MEC} is:

$$f_{ij}^{MEC*} = \sqrt{\frac{\lambda_i^T F_i}{\beta^*}}, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \quad (22)$$

$\beta^* > 0$, the objective function tends to reach its optimal point, then:

$$\sum_{j \in \mathcal{M}} f_{ij}^{MEC} - F_{j,max}^{MEC-L} = 0, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \quad (23)$$

Integrating (22) into (23), the lagrange multiplier becomes:

$$\beta^* = \frac{\sum_{j \in \mathcal{M}} \sqrt{\lambda_i^T F_i}}{(F_{j,max}^{MEC-L})^2}, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \quad (24)$$

Finally, by bringing (24) to (22), the equation for optimal resource allocation is written:

$$f_{ij}^{MEC*} = \frac{F_{j,max}^{MEC-L} \sqrt{\lambda_i^T F_i}}{\sum_{j \in \mathcal{M}} \sqrt{\lambda_i^T F_i}}, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M} \quad (25)$$

2.3. Federated dual-aggregation MetaL within the MEC framework (FedMeta2Ag)

In this subsection, we introduce an algorithm, FedMeta2Ag, designed to address the trade-off between training model performance, total time, and mobile device energy consumption. It is important to note that MetaL allows for optimizing performance based on a small number of adaptation examples across diverse tasks.

This algorithm is increasingly utilized within the domain of supervised learning and reinforcement learning [34]. The MAML is a gradient-based MetaL algorithm that operates through two interconnected stages: meta-training and meta-testing. During meta-training, a flexible initial model is learned to facilitate quick adaptation across various tasks, while meta-testing involves adapting the initial model to a specific task.

2.3.1. Description of the FedMeta2Ag algorithm

In the FedMeta2Ag algorithm, each IoT device leverages the MetaL approach on its respective dataset $D_i = \{(x_i^{di}, y_i^{di})\}$ containing input x_i^{di} and label y_i^{di} .

During each episode, a task T is randomly sampled from a task distribution $P(T)$ over D^{tr} , where in the meta-training involves a support set D_S^{tr} and a query set D_Q^{tr} . Each local meta-model is initialized with the server's θ parameters and subsequently trained using D^{tr} .

Specifically, each IoT device d_i downloads the parameter θ from the MEC-L server for initialization and then trains the model f_θ on the support set D_S^{tr} using one or two gradient descent steps. The training loss is determined by the error of the base model f_θ (the base-learner in MetaL), and is obtained by:

$$L_{D_S^{tr}} := \frac{1}{D_S^{tr}} \sum_{(x_i^{di}, y_i^{di}) \in D_S^{tr}} l(f_\theta(x_i^{di}), y_i^{di}) \quad (26)$$

In the evaluation steps (outer loop), it tests the base model f_θ on the query set D_Q^{tr} to update the parameters from θ^{di} at $\theta^{di'}$. As a result, the loss $L_{D_Q^{tr}}$ is calculated to update the base model.

$$L_{D_Q^{tr}} := \frac{1}{D_Q^{tr}} \sum_{(x_i^{di}, y_i^{di}) \in D_Q^{tr}} l(f_\theta(x_i^{di}), y_i^{di}) \quad (27)$$

Algorithm 1 FedMeta2Ag with double aggregation (**FedMeta2Ag**)**Require:** Task distribution: $P(T)$, Hyperparameters: α, β, k, m **Ensure:** Aggregate model

```

1: //Execution on Central Server
2: Update global model
3: Initialize  $\theta_c$  for each MEC-L server
4: for each MEC-L server  $m_l \in \mathcal{M}$  do
5:   Retrieve  $\theta_l$  from each MEC-L server
6:    $\theta_c = \frac{1}{m_l} \sum \theta_l$ 
7: end for
8: //Execution on Local Server
9: Update global model for each device
10:  $\theta \leftarrow \theta_c$ 
11: Initialize  $\theta$  for each device
12: for each episode  $E = e_1, e_2, \dots, e_i$  do
13:   Sample of  $m$  devices and distribution of  $\theta$ 
14:   for each device  $d_i \in D_i$  in parallel do
15:      $g_u \leftarrow \text{ModelTrainedMAML}(\theta)$ 
16:   end for
17:    $\theta \leftarrow \theta - \frac{\beta}{m} \sum_{d_i \in D_i} g_u$ 
18: end for
19: //Execution on IoT device
20:  $\text{ModelTrainedMAML}(\theta)$ 
21: Sample the data set in support  $D_S^{tr}$  and query  $D_Q^{tr}$ 
22: Calculate the loss  $L_{D_S^{tr}}$  using (26)
23:  $\theta^{di} \leftarrow \theta - \alpha \nabla_{\theta} L_{T_k}(f_{\theta})$ 
24: Calculate loss  $L_{D_Q^{tr}}$  using (27)
25:  $\theta^{di'} \leftarrow \theta^{di} - \beta \nabla_{\theta} L_{T_k}(f_{\theta} di)$ 
26: Find the decision to unload tasks  $X^*$ 
27: Obtain a high-performance model for the central server
28:  $g_u \leftarrow \theta^{di'}$ 

```

It then sends the $f\theta^{di'}$ meta-model to the MEC-L server. The MEC-L server aggregates the various local meta-models to update the global model. Considering that each MEC-L server is only responsible for a limited number of IoT devices, each MEC-L server obtains its own meta-model, whose performance may differ from that obtained by other servers. All these meta-models are sent to a MEC-C server, which aggregates them into a more general model. This more general model is sent to all MEC-L servers, which dispatch the new initialization parameters to all IoT devices. The learning rate of the inner loop is defined by α and that of the outer loop by β . In the meta-test, the decision is the test set D^{tr} , also composed of a support set D_S^{tr} and a query set D_Q^{tr} .

2.3.2. FedMeta2Ag integrating reinforcement learning

The proposed algorithm, FedMetaR2Ag, integrates reinforcement learning, specifically proximal policy optimization [35]. It is proposed to jointly manage task offloading and resource allocation decisions in the MEC environment. The objective is to quickly determine the optimal task offloading decision X^* and compute the optimal computational resource allocation strategy F^* based on the KKT conditions using the MAML algorithm. Each task T_k has an initial state distribution $q_i(X_1)$, transition distribution $q_i(X_{t+1}|X_t, a_t)$, and a loss corresponding to the reward function.

The fundamental components of reinforcement learning include the environment, agent, state, action, and reward. In our system, the controller acts as the agent and interacts with the MEC environment. For problem Q_1 , we define the state, action, and reward as:

- State: the state ($s(t)$) serves as the basis for the agent's action selection [36]. In the MEC system, the agent needs to acquire specific states.

- Action: once the state $s(t)$ is obtained, the reinforcement learning policy π is integrated into the network, enabling the network to choose the optimal task offloading action to maximize the system's reward. In this system, the agent's action, referred to as task unloading strategy X is responsible for offloading tasks [37].

$$a(t) = [x_1(t), x_2(t), \dots, x_N(t)] \quad (28)$$

- Reward: the main aim of the system is to minimize the objective function $Q(X, F)$, as part of its optimization process, while reinforcement learning is employed to maximize the system's reward.

The combination of reinforcement learning and MetaL enables the creation of a meta-policy f_θ that can be efficiently adapted to new tasks T . Here θ represents the parameter of the policy network π . Initially, K tasks T_k are randomly sampled from the task distribution $P(T)$, where $k \in [1, K]$. T_k , M trajectories D_k are generated using f_θ . These trajectories D_k are then utilized to update the sub-meta policy f'_θ .

$$\theta'_k = \theta' - \alpha \nabla \theta' L_{T_k}(f'_\theta) \quad (29)$$

In this context α represents the learning rate, which determines the rate at which the model adapts to new information. L_{T_k} refers to the loss function associated with task T_k , which is defined based on the reward obtained from the reinforcement MetaL model.

$$L_{T_k}(f_\phi) = -\mathbb{E}_s(t), a(t) \sim f_\phi \left[\sum_{t=1}^H r_k(s(t), a(t)) \right] \quad (30)$$

In this scenario, H represents the horizon of the markov decision process, which determines the length of the decision-making sequence, f_ϕ refers to the model for which the loss is to be calculated.

Once the f'_θ sub-meta policy has been updated, it is utilized to sample a trajectory of task T_k . This trajectory is then used to calculate the valid loss $L_{T_k}(f'_{\theta_k})$ of the sub-meta policy f'_{θ_k} . The meta-policy update is defined as, with β representing the meta-policy rate:

$$\theta = \theta - \beta \nabla_\theta L_{T_k}(f_{\theta_k}) \quad (31)$$

when a new task that conforms to the task distribution $P(T)$ is encountered, it has the potential to achieve convergence with fewer gradient updates. In this context, η represents the learning rate, which determines the step size of the gradient updates during the learning process.

$$\theta_{fin} = \theta - \eta \nabla_\theta L_T(f_\theta) \quad (32)$$

The task unloading decision, being a discrete variable, is addressed using a discrete proximal policy optimization (PPO) and a meta-reinforcement learning framework called a MAML. The resource allocation policy is computed using KKT conditions, and FL is employed to aggregate the models obtained from each device. These three algorithms are combined to form the FedMetaR2Ag algorithm, which is described in detail in Algorithm 2 (in Appendix).

3. ANALYSIS OF RESULTS AND DISCUSSION

In this section, we evaluate the performance of our system, joint resource management for MEC-IoT environment based on meta-FL, and discuss the obtained results. The key performance criteria used in the measurements include learning accuracy and losses. Additionally, we evaluate the efficiency of our algorithm by considering its learning time for different scenarios.

3.1. Simulation parameters settings

To simulate the proposed FedMeta2Ag algorithm, we used the Anaconda 2021 development environment (version 4.12.0) and the Python programming language version 3.9.12.

The hardware used for the simulation was an 8th generation HP ProBook 430 G5 computer with an i5-8250U CPU running at 1.80 GHz, 8 GB of RAM, and a 257 GB SSD.

For the evaluation of the algorithm, we used the Modified National Institute of Standards and Technology (MNIST) database. We randomly selected 50% of the data for the training phase, while the remaining

data was used for the test phase. Each IoT device divided its local dataset into a support set and a query set. The support set contained only one labeled example for each class.

To showcase the efficiency of the FedMeta2Ag algorithm in MEC server scenarios, we conducted multiple tests and present the simulation results to verify its effectiveness.

For our evaluation, we considered batches of 50, 100, 150, and 200 customers. The number of epochs varied between 5, 10, 15, and 20. It is important to note that all other simulation parameters were kept constant during these experiments.

3.2. Analysis of accuracy results

We evaluate the accuracy of our system on the basis of different epochs (5, 10, 15, and 20), during the training and test phase, considering 50, 100, 150, and 200 customers. By observing the evolution of accuracy measurements over time, we can evaluate the learning performance and convergence of our algorithm FedMeta2Ag.

3.2.1. Accuracy considering 5 epochs

The accuracy results obtained after 5 training epochs, for 50 iterations, varied based on the batch size. With a batch size of 50 customers, the training accuracy (see Figure 2) reached 91.4%, and the testing accuracy (see Figure 3) reached 91.9%. When the batch size was increased to 100, 150, and 200 customers, the training and test accuracies were 89.5% and 90.3%, 88.4% and 89.2%, 87.2% and 88.0%, respectively.

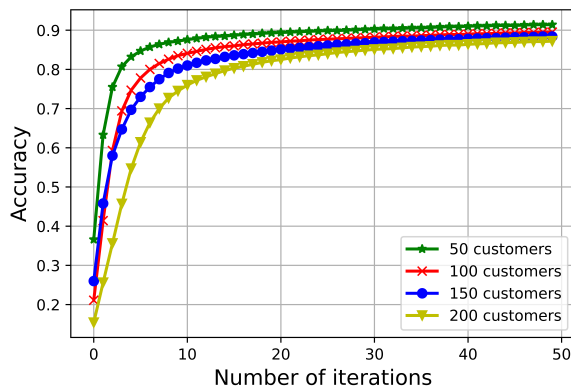


Figure 2. Training phase considering model accuracy at 5 epochs

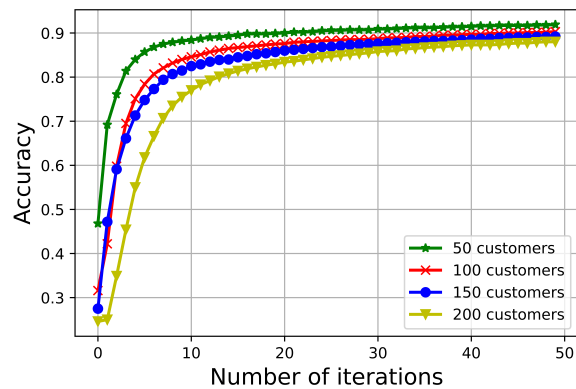


Figure 3. Test phase considering model accuracy at 5 epochs

We can be concluded that the optimal results for the majority of customers are attained after the tenth (10^{th}) training stage. However, as the number of customers increases, the model's performance progressively declines. Moreover, the simulation results show that the 200-customer batch requires a minimum of 15 training steps to achieve optimal results as compared to other simulation cases. On the other hand, the results for the 50-customer batch are better than those for other customers, indicating that its meta-model is easier to train and exhibits high stability.

3.2.2. Accuracy considering 10 epochs

The accuracy results obtained after 10 training epochs, for 50 iterations, varied based on the batch size. With a batch size of 50 customers, the training accuracy (see Figure 4) reached 91.6% and the testing accuracy (see Figure 5) reached 91.9%. When the batch size was increased to 100, 150, and 200 customers, the training and test accuracies were 89.7% and 90.3%, 88.3% and 88.6%, 87.5% and 88.4%, respectively.

By varying the epoch to 10 while keeping the same batch of customers, we notice that starting from the 27^{th} iteration, the performance achieved reaches a plateau of 90%. This performance remains relatively constant, with a slight improvement to 91% from the 39^{th} iteration onwards. However, as we increase the number of customers, we consistently observe that the system becomes saturated, and the model's performance declines. This suggests that there is a limit to the system's capacity to handle a higher number of customers, resulting in decreased performance.

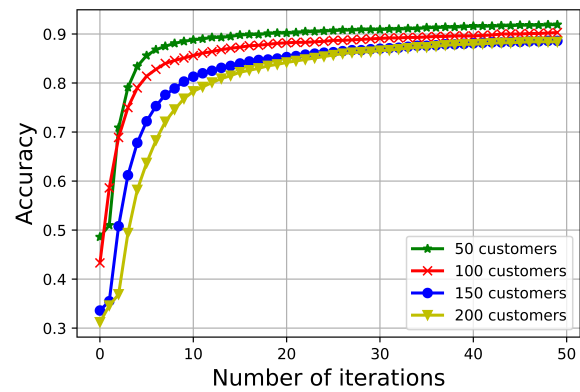
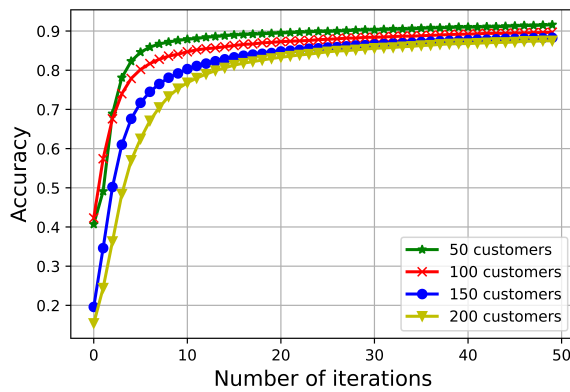


Figure 4. Training phase considering model accuracy at 10 epochs

Figure 5. Test phase considering model accuracy at 10 epochs

3.2.3. Accuracy considering 15 epochs

The accuracy results obtained after 15 training epochs, for 50 iterations, varied based on the batch size. With a batch size of 50 customers, the training accuracy (see Figure 6) reached 91.7% and the testing accuracy (see Figure 7) reached 91.8%. When the batch size was increased to 100, 150, and 200 customers, the training and test accuracies were 89.6% and 90.3%, 88.6% and 88.6%, 87.3% and 88.4%, respectively. This indicates that the model performs better in the test phase compared to the training phase, suggesting that it is able to generalize well to unseen data.

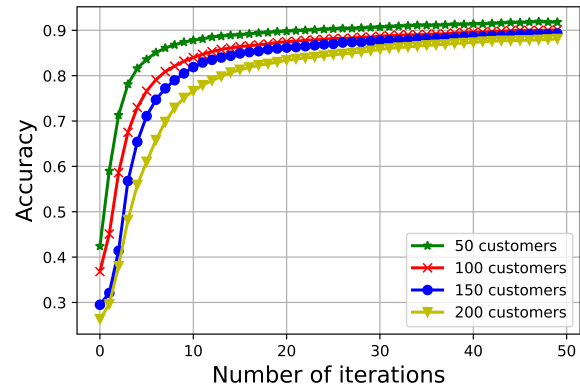
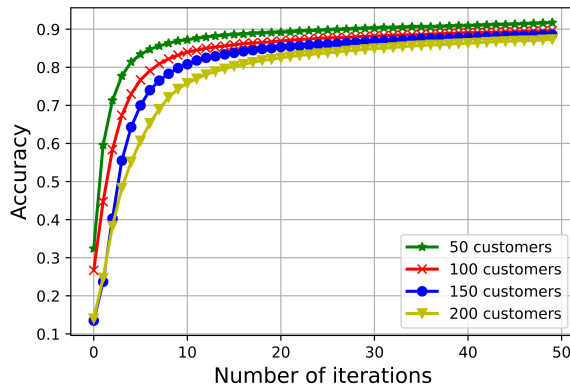


Figure 6. Training phase considering model accuracy at 15 epochs

Figure 7. Test phase considering model accuracy at 15 epochs

3.2.4. Accuracy considering 20 epochs

The accuracy results obtained after 20 training epochs, for 50 iterations, varied based on the batch size. With a batch size of 50 customers, the training accuracy (see Figure 8) reached 91.5% and the testing accuracy (see Figure 9) reached 92.0%. When the batch size was increased to 100, 150, and 200 customers, the training and test accuracies were 89.6% and 90.1%, 88.5% and 89.3%, 87.5% and 88.4%, respectively.

As we consider increasingly larger batches of customers, we observe a decline in the model's performance. With a batch of 50 customers, the model's performance starts at 80% and gradually increases to 91.5%. For a batch of 100 customers, the best performance achieved is 89.6%. With a batch of 150 customers, the model's performance is found to be 88.5%. Finally, with 200 customers, we obtain a performance of 87.5% and 88% in the training and test phases respectively.

These results suggest that the performance of the system is limited by its capacity to handle large batches of customers. As we overload the system with more customers, its ability to perform effectively is

reduced, ultimately leading to a decline in performance. Therefore, it is essential to find the optimal batch size that balances performance and capacity, in order to ensure the best possible results for the task at hand.

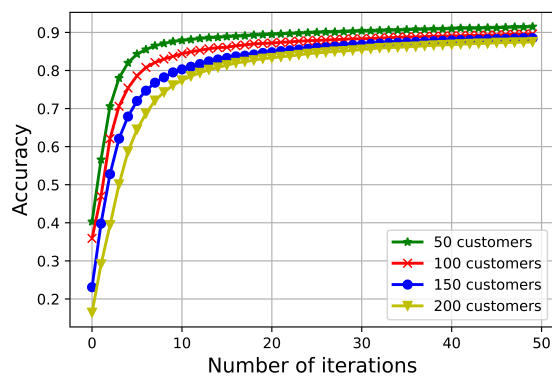


Figure 8. Training phase considering model accuracy at 20 epochs

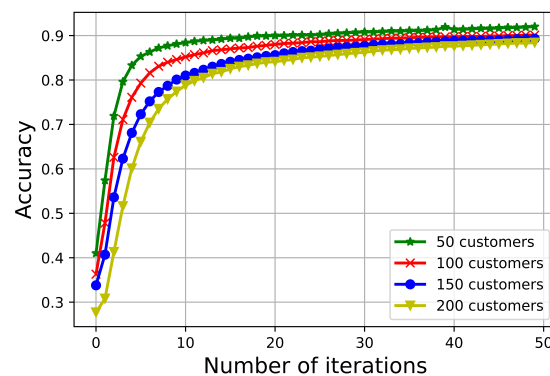


Figure 9. Test phase considering model accuracy at 20 epochs

3.2.5. Summary

In all cases of simulation, we can see that the performance obtained in the test phase is far superior to that obtained in the training phase. Considering 20 epochs, performance during training is 91.4%, while that obtained with test data is 92.0%. Performance continues to increase during the first 20 iterations, then becomes progressively stable. We note that epoch variation has no impact on the model, as the aim is to have a high-performance model that can adapt to different tasks. The MAML algorithm allows rapid generalization, so we don't need to go to several epochs to decide. With just 5 epochs we can already decide. As we increase the number of customers, model performance decreases in almost all cases. As illustrated by Figure 10, performance with 50 customers is better than with 100, 150, and 200 customers. It's only the variation in the number of customers that impacts on the algorithm's performance. When there are several customers, the system is overloaded and performance gradually deteriorates. The proposed model only takes into account a limited number of devices. Observations from the local meta-models of each IoT device indicate that performance consistently improves during the initial 10 steps and then stabilizes gradually. Since each meta-model requires five local iterations, the model achieves convergence within 50 rounds. The results depicted in the various figures demonstrate that the FedMeta2Ag algorithm with double aggregation (FedMeta2Ag) consistently outperforms other approaches, exhibiting faster and more stable convergence.

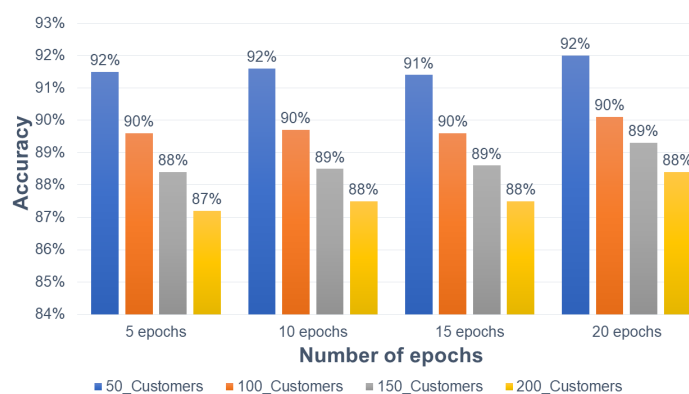


Figure 10. Observation of performance by varying epochs by considering 50, 100, 150, and 200 customers

3.3. Analysis of losses results

In section 3.1, our algorithm is evaluated in terms of accuracy, considering different epochs (5, 10, 15, and 20) for a simulation of 50 iterations. Our algorithm is now evaluated in terms of losses for a training of 5,

10, 15, and 20 simulated at 50 iterations.

In the subsections, we examine the losses obtained during the training and test phases for different batch sizes and epoch numbers. Specifically, we analyze the losses obtained for the batch sizes of 50, 100, 150, and 200 customers. For each of these batch sizes, we vary the number of epochs between 5, 10, 15, and 20. We keep all other simulation parameters constant.

By analyzing the losses during the training and test phases, we can obtain insights into the effectiveness of the model for different batch sizes and epoch numbers. These insights can help us determine the optimal combination of batch size and epoch number that yields the best results for the task at hand.

3.3.1. Losses considering 5 epochs

The loss results obtained after 5 learning epochs, for 50 iterations, varied according to batch size. Considering 50 customers in the training phase, we observe that the loss decreases and reaches 30.3% (Figure 11). While it decreases during the test phase and reaches its lowest point at 29.1% (Figure 12). With 100 customers, the loss decreases to 38.8% in training and 36.0% in the test phase. Observing the Figure 12, the losses are a bit higher than the previous two cases. They decrease to 45.5% and 43.4% respectively in the training and test phase with 150 customers, to 52.8% in the training phase and 49.8% in the test phase with 200 customers.

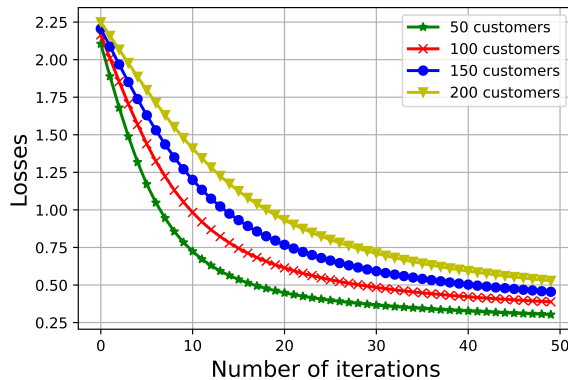


Figure 11. Model losses at training phase considering 5 epochs

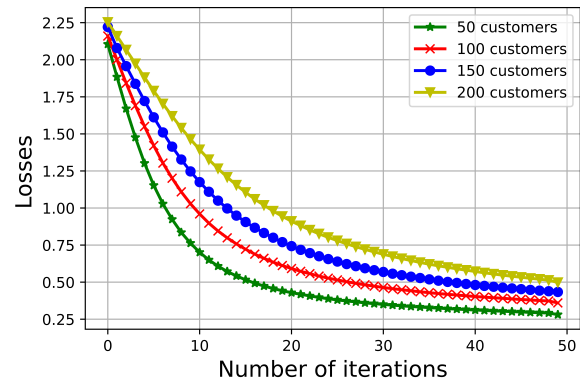


Figure 12. Model losses at test phase considering 5 epochs

3.3.2. Losses considering 10 epochs

The loss results obtained after 10 learning epochs, for 50 iterations, varied according to batch size. With a batch size of 50 customers, the training loss reached 30.2% and the test loss reached 29.0%. When the batch size was increased to 100, 150, and 200 customers, the training and test losses were 38.4% and 36.4%, 45.4% and 43.2%, 52.6% and 50.0% respectively. By increasing the number of customers, the losses in the training and test phase increase respectively (see Figures 13 and 14).

3.3.3. Losses considering 15 epochs

In this part, we set this time the number of epochs to 15. We clearly observe that the more the number of customers increases, the more the losses increase. This therefore degrades the performance of the model in the training phase and in the test phase.

Thus, we find that the training losses for 50 customers are 30.4% (see Figure 15) and 28.9% in the test phase (see Figure 16). They increase to 38.8% and 36.4% respectively in training and testing with 100 customers. With 150 customers, the losses are 45.6% in the training phase and 43.0% in the test phase. The losses continue to increase and reach 52.8% and 50.0% respectively in the training and test phase with 200 customers.

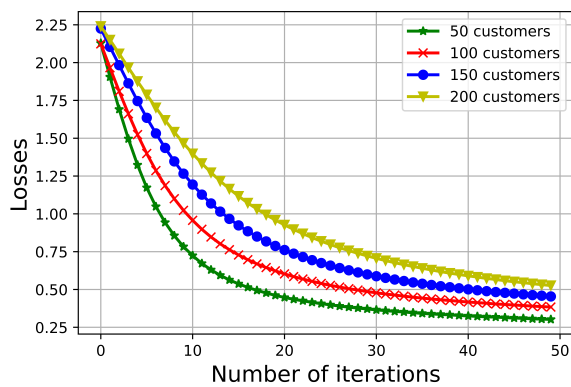


Figure 13. Model losses at training phase considering 10 epochs

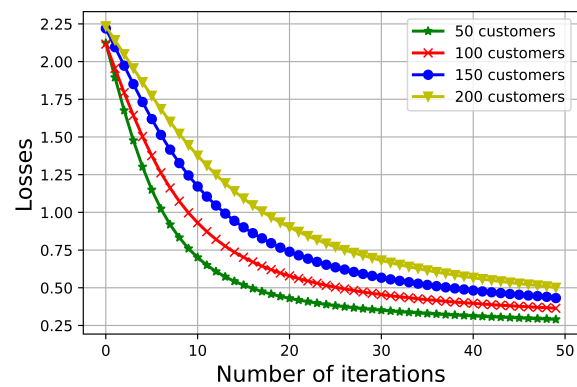


Figure 14. Model losses at test phase considering 10 epochs

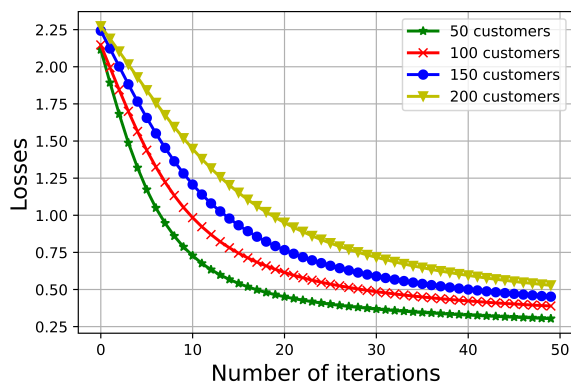


Figure 15. Model losses at training phase considering 15 epochs

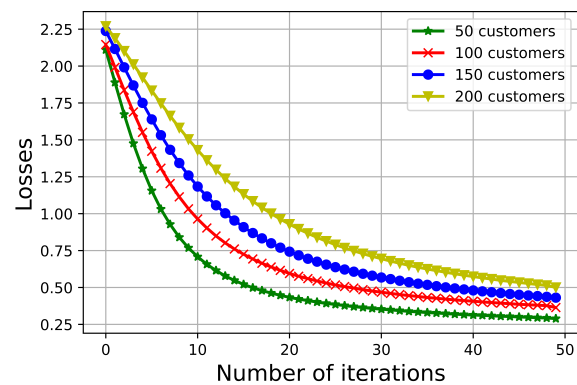


Figure 16. Model losses at test phase considering 15 epochs

3.3.4. Losses considering 20 epochs

The loss results obtained after 15 learning epochs, for 50 iterations, varied according to batch size. With a batch size of 50 customers, the training loss (see Figure 17) reached 30.2% and the test loss (see Figure 18) reached 28.6%.

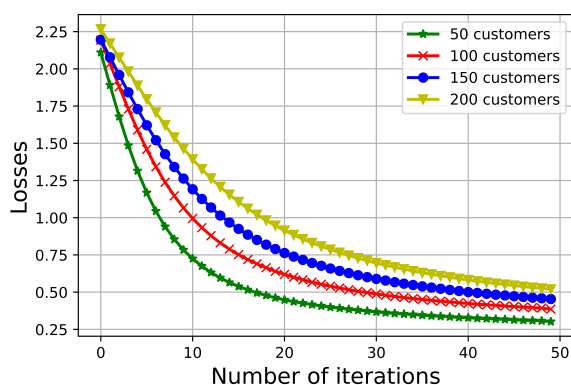


Figure 17. Model losses at training phase considering 20 epochs

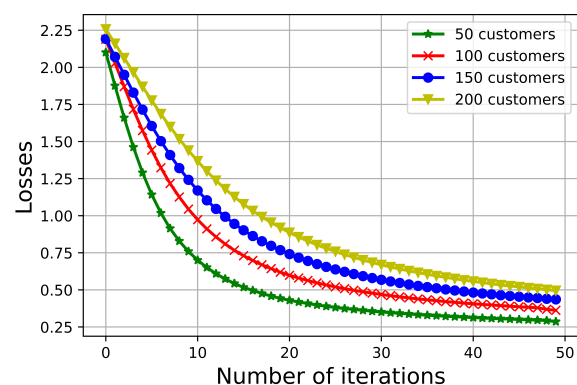


Figure 18. Model losses at test phase considering 20 epochs

Considering 20 epochs, the losses are around 30.2% and 28.6% in the training and test phase. With 100 customers, the losses obtained in the training phase are 38.6% and 36.2% in the test phase. Considering this time 150 customers, the losses are 45.4% and 42.8% respectively in the training and test phase. They increase with the number of customers and reach 51.9% and 49.5% respectively in the training and test phase with 200 customers.

3.3.5. Summary

The results of the training phase show that there was a loss of 30.3%, and in the subsequent test phase, the loss was reduced to 29.1%. This indicates that the model improved slightly in its ability to predict and minimize losses. However, it is worth noting that the number of customers had a direct impact on the losses in both phases. As the number of customers increased, the losses also increased. Figure 19 provides visual representation and additional insights on this relationship between the number of customers and losses in both the training and test phases.

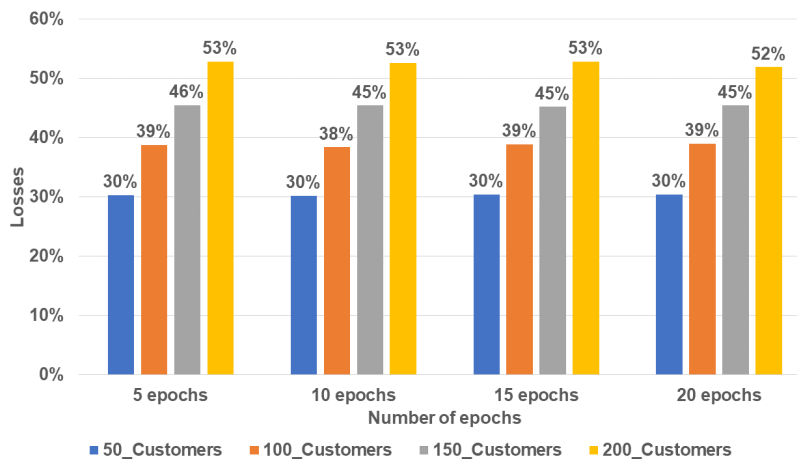


Figure 19. Observation of losses by varying epochs and customers

The obtained losses during training and testing are displayed in Table 1, providing a visual representation and additional insights on the relationship between the number of customers and losses. The table clearly illustrates the impact of the number of customers on both phases of the training and testing, which further supports the previously made observations. As the number of customers increases, there is a corresponding increase in losses in both phases of the training and testing. This relationship is consistent with the findings discussed earlier.

Table 1. Different simulation stages with 50 turns and 20 epochs

Number customers	Acc. train (%)	Ac. test (%)	Loss train (%)	Loss test (%)
50	91.5	92.0	30.4	29.1
100	89.6	90.1	39.0	37.3
150	88.5	89.3	45.4	43.6
200	87.5	88.4	51.9	49.5

3.3.6. Algorithm training time

In this part, we present the running time of the algorithm, as shown in Figure 20 considering 50, 100, 150, and 200 customers at 5 epochs. This time takes into account the time for the formation of the meta-model, that is to say the time between the device and the aggregation server. This observation is made by considering 20 iterations.

3.3.7. Overall execution time

This section deals with the time consumption of the model by varying the number of customers. As the histogram shows in Figure 21, the time consumption with 50 customers is much less than that of other batches of customers. Due to system overload, training with 50 customers consumes less, which is in with the

expectations. It can be observed that as the number of customers increases, the algorithm requires more time to compute the different training rounds.

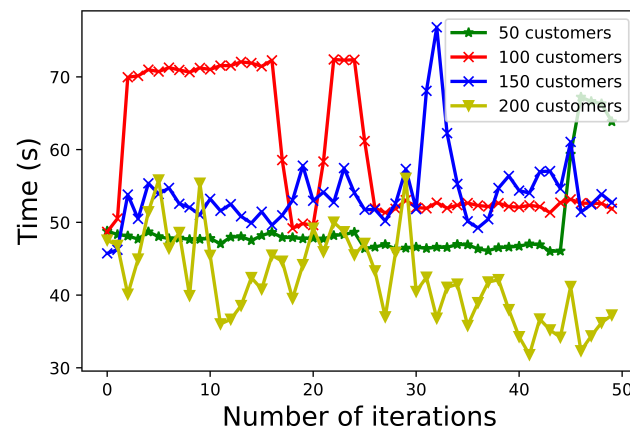


Figure 20. Training time by varying the number of customers

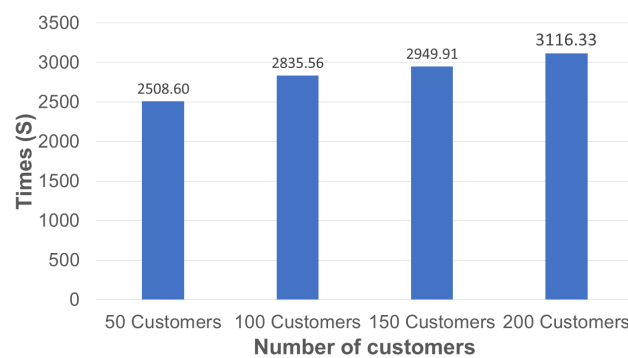


Figure 21. Overall execution time by considering 5 epochs and 50, 100, 150, and 200 customers

Figure 21 presents the overall execution time analysis, considering 5 epochs and varying customer numbers (50, 100, 150, and 200). The figure provides a visual representation of the relationship between the number of customers and the overall execution time. The time consumption with 50 customers is much less than that of other batches of customers. It can be observed that as the number of customers increases, the algorithm requires more time to compute the different training rounds.

3.4. Discussion

Hangjia *et al.* [38] proposed a privacy-preserving algorithm for client selection based on the double deep Q-networks (DDQN) algorithm and FL. Experiments were carried out on different FL server tasks. The simulation results give an accuracy of 89%. The simulation was run on 250 rounds using the fashion-MNIST database. Whereas with FedMeta2Ag, the algorithm achieves an accuracy of 91% with only 5 epochs. MAML enables rapid adaptation of FL. However, the authors did not evaluate the algorithm's losses.

Li *et al.* [39] investigated a task offloading mechanism in MEC using federated reinforcement learning. They evaluated the effectiveness of the algorithm by analyzing the model loss under different learning rates to determine the change in learning loss. The authors found that when the learning rate was set to 0.1 and 0.0001, the training loss was relatively high. However, when the learning rate was set to 0.01, the loss decreased earlier, starting at 55% and eventually reaching 15%. This suggests that a learning rate of 0.01 was more effective in reducing the loss during training.

Furthermore, the results indicated that the model quality was influenced by the number of users. With more users participating, the model quality tended to be higher. However, the authors observed that the quality

of the overall model did not continue to improve with 30 participating users. The quality of the final model was found to be similar to that of the models with 12 and 20 users. Therefore, the number of users did not have a significant effect on the model quality.

In our case, the losses decrease to 29%. With fewer iterations, the model loss was greater, while with more iterations, the results gradually stabilized, which aligns with our theory. By increasing the number of iterations and maintaining the same environment, our algorithm could potentially achieve better results compared to the findings in [39]. This difference can be explained by the fact that [39] uses reinforcement learning, which learns with rewards. In our case, MetaL learns from another algorithm and with few epochs, the algorithm is stable.

Li *et al.* [40] proposed an algorithm for federated reinforcement learning (q-FedAvg) in edge networks. They used several types of data to confirm the model's effectiveness. Using the synthetic database, the performance of the model obtained with 2000 training rounds is 80%. Our model, on the other hand, performed better than 90%.

Based on this observation, we claim that the proposed model demonstrates higher performance and is well-suited for MEC, validating the integrated model in a FL framework with two levels of aggregation. We have achieved a scalable and computationally efficient framework that eliminates the requirement for centralized computation. This framework offers flexibility in striking a balance between communication cost and local computation cost.

4. CONCLUSION

This study presented a solution for task offloading and resource allocation in MEC for IoT by utilizing a FedMeta2Ag approach. The paper introduces the FedMeta2Ag algorithm, which leverages FedMeta2Ag to optimize task offloading and resource allocation decisions for IoT devices. The algorithm's effectiveness is assessed using the MNIST dataset, and the results demonstrate improved performance during the test phase compared to the training phase. One advantage of this approach is its ability to learn from the experiences of all IoT devices, resulting in more robust and accurate models. With a consideration of 20 epochs, the training accuracy is measured at 91.5%, while the test data achieves an accuracy of 92.0%. Furthermore, the accuracy steadily increases during the initial 20 iterations and eventually stabilizes. Additionally, the performance evaluation includes a comparison with existing methods, revealing that our approach predicts performance with greater accuracy than the existing models. This approach effectively meets the demanding performance requirements of wireless communication systems.

For our future work, we will consider further improvements in future research: i) scaling the approach: this article primarily concentrated on the MNIST dataset, which is considered relatively small. To further evaluate the performance and effectiveness of the FedMeta2Ag algorithm, future research can investigate its scalability to larger and more intricate datasets. This exploration will enable the algorithm to be assessed in real-world scenarios that involve a higher volume of data; ii) robustness towards heterogeneous IoT devices: this article assumes a uniformity of capabilities and characteristics among IoT devices, which may not reflect the reality of heterogeneous IoT environments. In practice, IoT devices can vary significantly in terms of processing power, energy constraints, and network connectivity. It is important for future research to consider the robustness of algorithms and approaches in accommodating such heterogeneity. This will ensure that the proposed solutions can effectively handle the diverse range of IoT devices encountered in real-world scenarios; iii) confidentiality and security considerations: confidentiality and security are crucial considerations in FL, as it involves training models on decentralized data from multiple devices. To ensure the protection of sensitive data and preserve user privacy, it is important for future research to focus on enhancing the privacy and security mechanisms within the FedMeta2Ag algorithm. This can involve exploring techniques such as encryption, differential privacy, secure aggregation, and secure communication protocols. By addressing these concerns, the algorithm can provide stronger safeguards for data confidentiality and user privacy during the training process; and iv) real-world deployment and validation: this article conducted performance evaluations of the FedMeta2Ag algorithm using simulated scenarios. To gain a deeper understanding of the practical challenges and benefits associated with implementing the algorithm in real-world MEC-IoT environments, future research can focus on its deployment and validation in such settings. This real-world deployment and validation will provide valuable insights into the algorithm's effectiveness and feasibility in practical scenarios, helping to bridge the gap between theoretical evaluations and real-world applications.

APPENDIX

Algorithm 2 FedMeta2Ag incorporating reinforcement learning (**FedMetaR2Ag**)**Require:** Task distribution: $P(T)$, Hyperparameters: $\alpha, \beta, \eta, k, m$ **Ensure:** Optimal decision for task unloading X^* and resource allocation F^* .

```

1: //Execution on Central Server
2: Update global model
3: Initialize  $\theta_c$  for each MEC-L server
4: for each MEC-L server  $m_l \in \mathcal{M}$  do
5:   Retrieve  $\theta_l$  from each MEC-L server
6:    $\theta_c = \frac{1}{m_l} \sum \theta_l$ 
7: end for
8: //Execution on Local Server
9: Update global model for each device
10:  $\theta \leftarrow \theta_c$ 
11: Initialize  $\theta$  for each device
12: for each episode  $E = e_1, e_2, \dots, e_i$  do
13:   Sample of  $m$  devices and distribution of  $\theta$ 
14:   for each device  $d_i \in D_i$  in parallel do
15:      $g_u \leftarrow \text{trainedMAMLmodel}(\theta)$ 
16:   end for
17:    $\theta \leftarrow \theta - \frac{\beta}{m} \sum_{d_i \in D_i} g_u$ 
18: end for
19: //Execution on IoT device
20: ModelTrainedMAML( $\theta$ )
21: while not done do
22:   From the distribution  $P(T)$  of the sample of  $k$  tasks:  $P(T) \sim T_k$ 
23:   for  $T_k$  do
24:     Sample  $K$  trajectories  $D_k$  of  $T_k$  using  $f_{\theta}$  and calculate the loss  $L_{T_k}(f_{\theta'})$ : equation (30)
25:      $\theta'_k = \theta - \alpha \nabla \theta' L_{T_k}(f_{\theta'})$ 
26:     Sample a trajectory  $D'$  using  $f_{\theta'_k}$  and calculate the loss  $L_{T_k}(f_{\theta'_k})$ : (27)
27:   end for
28:   Update meta-policy  $\theta = \theta - \beta \nabla_{\theta} L_{T_k}(f_{\theta'_k})$ 
29: end while
30: Sample a new task  $P(T) \sim T$  and  $K$  trajectories  $D$  using  $f_{\theta}$ 
31: Update final policy  $\theta_{fin} = \theta - \eta \nabla_{\theta} L_T(f_{\theta})$ 
32: Find the decision to unload tasks  $X^*$ .
33: Calculate the optimal allocation solution  $F^*$  using (25)
34:  $g_u \leftarrow \theta_{fin}$ 
35: Return  $g_u$  to MEC-L server

```

REFERENCES




- [1] Q.-V. Pham *et al.*, "A Survey of Multi-Access Edge Computing in 5G and Beyond: Fundamentals, Technology Integration, and State-of-the-Art," *IEEE Access*, vol. 8, pp. 116974–117017, 2020, doi: 10.1109/access.2020.3001277.
- [2] G. Nencioni, R. G. Garroppo and R. F. Olimid, "5G Multi-Access Edge Computing: A Survey on Security, Dependability, and Performance," in *IEEE Access*, vol. 11, pp. 63496–63533, 2023, doi: 10.1109/ACCESS.2023.3288334.
- [3] I. Dias, L. Ruan, C. Ranaweera, and E. Wong, "From 5G to Beyond: Passive Optical Network and Multi-Access Edge Computing Integration for Latency-Sensitive Applications," *Optical Fiber Technology*, vol. 75, p. 103191, 2023, doi: 10.1016/j.yofte.2022.103191.
- [4] S. Kumar, P. Tiwari, and M. Zymbler, "Internet of Things is a revolutionary approach for future technology enhancement: a review," *Journal of Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0268-2.
- [5] X. Li, J. Zhang, and C. Pan, "Federated Deep Reinforcement Learning for Energy-Efficient Edge Computing Offloading and Resource Allocation in Industrial Internet," *Applied sciences*, vol. 13, no. 11, pp. 6708–6708, May 2023, doi: 10.3390/app13116708.
- [6] A. Sarah, G. Nencioni, and Md. M. I. Khan, "Resource Allocation in Multi-access Edge Computing for 5G-and-beyond networks," *Computer Networks*, p. 109720, Mar. 2023, doi: 10.1016/j.comnet.2023.109720.

- [7] S. Yue, J. Ren, J. Xin, D. Zhang, Y. Zhang, and W. Zhuang, "Efficient Federated Meta-Learning Over Multi-Access Wireless Networks," in *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 5, pp. 1556-1570, May 2022, doi: 10.1109/JSAC.2022.3143259.
- [8] M. Al-Quraan et al., "Edge-Native Intelligence for 6G Communications Driven by Federated Learning: A Survey of Trends and Challenges," in *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 7, no. 3, pp. 957-979, June 2023, doi: 10.1109/TETCI.2023.3251404.
- [9] X. Liu, Y. Deng, A. Nallanathan, and M. Bennis, "Federated Learning and Meta Learning: Approaches, Applications, and Directions," in *IEEE Communications Surveys and Tutorials*, doi: 10.1109/COMST.2023.3330910.
- [10] S. Trindade, L. F. Bittencourt, and N. L. S. da Fonseca, "Resource management at the network edge for federated learning," *Digital Communications and Networks*, Oct. 2022, doi: 10.1016/j.dcan.2022.10.015.
- [11] Z. Tong, X. Deng, F. Ye, S. Basodi, X. Xiao, and Y. Pan, "Adaptive computation offloading and resource allocation strategy in a mobile edge computing environment," *Information Sciences*, vol. 537, pp. 116-131, Oct. 2020, doi: 10.1016/j.ins.2020.05.057.
- [12] M. Moshawrab, M. Adda, A. Bouzouane, H. Ibrahim, and A. Raad, "Reviewing Federated Learning Aggregation Algorithms; Strategies, Contributions, Limitations, and Future Perspectives," *Electronics*, vol. 12, no. 10, p. 2287, Jan. 2023, doi: 10.3390/electronics12102287.
- [13] K. Moghaddasi and S. Rajabi, "Double Deep Q-Learning Networks for Energy-Efficient IoT Task Offloading in D2D MEC Environments," *2023 7th International Conference on Internet of Things and Applications (IoT)*, Isfahan, Iran, Islamic Republic of, 2023, pp. 1-6, doi: 10.1109/IoT60973.2023.10365356.
- [14] X. Yang, X. Yu, H. Huang, and H. Zhu, "Energy Efficiency Based Joint Computation Offloading and Resource Allocation in Multi-Access MEC Systems," *IEEE Access*, vol. 7, pp. 117054-117062, 2019, doi: 10.1109/access.2019.2936435.
- [15] J. Wang, Y. Wang, and H. Ke, "Joint Optimization for MEC Computation Offloading and Resource Allocation in IoV Based on Deep Reinforcement Learning," *Mobile Information Systems*, vol. 2022, pp. 1-11, Aug. 2022, doi: 10.1155/2022/9230521.
- [16] A. Dandoush, A. Gouisssem, and B. Ciftler, "Secure and Privacy-Preserving Federated Learning-Based Resource Allocation for Next Generation Networks," *INDIGO (University of Illinois at Chicago)*, Apr. 2023, doi: 10.36227/techrxiv.22591852.v1.
- [17] M. Schwind, "Dynamic pricing and automated resource allocation for complex information services: Reinforcement learning and combinatorial auctions," *Springer Science and Business Media*, 2009, doi: 10.1007/978-3-540-68003-1.
- [18] S. Shakyia and S. Shakyia, "Resource Allocation and Power Management in Cloud Servers Using Deep Reinforcement Learning," *Advances in intelligent systems and computing*, pp. 789-798, Oct. 2021, doi: 10.1007/978-981-16-5157-1_62.
- [19] J. Wang, "System Optimisation for Multi-access Edge Computing Based on Deep Reinforcement Learning," *University of Exeter (United Kingdom)*, 2021.
- [20] Y. Li, J. Li, and J. Pang, "A Graph Attention Mechanism-Based Multiagent Reinforcement-Learning Method for Task Scheduling in Edge Computing," *Electronics*, vol. 11, no. 9, p. 1357, Apr. 2022, doi: 10.3390/electronics11091357.
- [21] A. A. Abd Al-Ameer and W. S. Bhaya, "Federated learning security mechanisms for protecting sensitive data," *Bulletin of Electrical Engineering and Informatics*, vol. 12, no. 4, pp. 2421-2427, Aug. 2023, doi: 10.11591/eei.v12i4.4751.
- [22] B. A. Adoum, F. M. Fidel, S. Faustin, A. Moungache, and N. Armi, "Federated Meta-Learning for task offloading and resource allocation in MEC-IoT," *2023 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET)*, Bandung, Indonesia, 2023, pp. 122-128, doi: 10.1109/ICRAMET60171.2023.10366780.
- [23] Y. Wenjun, D. Pingping, C. Lin and T. Wensheng, "Loss-Aware Throughput Estimation Scheduler for Multi-Path TCP in Heterogeneous Wireless Networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 5, pp. 3336-3349, 2021, doi: 10.1109/TWC.2021.3049300.
- [24] C. Xing and L. Guizhong, "Federated Deep Reinforcement Learning-Based Task Offloading and Resource Allocation for Smart Cities in a Mobile Edge Network," *Sensors*, vol. 22, no. 13, 2022, doi: 10.3390/s22134738.
- [25] J. Feibo, D. Li, W. Kezhi, Y. Kun, and P. Cunhua, "Distributed resource scheduling for large-scale MEC systems: A multiagent ensemble deep reinforcement learning with imitation acceleration," *IEEE Internet of Things Journal*, vol. 9, no. 9, 2021, pp. 6597-6610, doi: 10.1109/IIOT.2021.3113872.
- [26] W. Jin, H. Jia, M. Geyong, Z. Albert Y, and G. Nektarios, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, 2020, pp. 242-253, doi: 10.1109/TPDS.2020.3014896.
- [27] Y. Chao, G. Yangshui, D. Hongwei, N. Rencan, and L. Bo, "Meta Reinforcement Learning Based Computation Offloading Strategy for Vehicular Networks," *IEEE Transactions on Parallel and Distributed Systems*, 2022, doi: 10.21203/rs.3.rs-1614949/v1.
- [28] H. Yixue, C. Min, H. Long, H. M. Shamim, and G. Ahmed, "Energy Efficient Task Caching and Offloading for Mobile Edge Computing," *IEEE Access*, vol. 6, pp. 11365-11373, 2018, doi: 10.1109/ACCESS.2018.2805798.
- [29] W. Kehao, H. Zhixin, A. Qingsong, Z. Yi, Y. Jihong, Z. Pan, C. Lin, and S. Hyundong, "Joint offloading and charge cost minimization in mobile edge computing," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 205-216, 2020, doi: 10.1109/OJCOMS.2020.2971647.
- [30] Y. Pengfei, C. Xin, C. Ying, and L. Zhuo, "Deep reinforcement learning based offloading scheme for mobile edge computing," *2019 IEEE International Conference on Smart Internet of Things*, pp. 417-421, 2019, doi: 10.1109/SmartIoT.2019.00074.
- [31] P. Shanchen and W. Shuyu, "Joint Wireless Source Management and Task Offloading in Ultra-Dense Network," *IEEE Access*, vol. 8, pp. 52917-52926, 2020, doi: 10.1109/ACCESS.2020.2980032.
- [32] W. Jyrki, D. James S, F. Peter C, S. Ralph E, Z. Stanley, and D. Kalyanmoy, "Multiple criteria decision making, multiattribute utility theory: Recent accomplishments and what lies ahead," *Management science*, vol. 54, no. 7, 2008, doi: 10.1287/mnsc.1070.0838.
- [33] P. Quoc-Viet, L. Tuan, T. Nguyen H, P. Bang Ju, and H. Choong Seon, "Decentralized computation offloading and resource allocation for mobile-edge computing: A matching game approach," *IEEE Access*, vol. 6, pp. 75868-75885, 2018.
- [34] J. Vanschoren, "Meta-Learning: A Survey," *ArXiv*, 2018, doi: 10.48550/arXiv.1810.03548.
- [35] D. Yueyue, X. Du, M. Sabita, and Z. Yan, "Joint Load Balancing and Offloading in Vehicular Edge Computing and Networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377-4387, 2019, doi: 10.1109/IIOT.2018.2876298.
- [36] R. Yijing, S. Yaohua, and P. Mugen, "Deep Reinforcement Learning Based Computation Offloading in Fog Enabled Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4978-4987, 2021, doi: 10.1109/TII.2020.3021024.




- [37] B. Arash, M. Setareh, T. Daniele, and H. Ekram, "Computation Offloading in Heterogeneous Vehicular Edge Networks: On-Line and Off-Policy Bandit Solutions," *IEEE Transactions on Mobile Computing*, vol. 21, no. 12, pp. 4233-4248, 2022, doi: 10.1109/TMC.2021.3082927.
- [38] Z. Hangjia, X. Zhijun, Z. Roozbeh, W. Tao, and C. Kewei, "Adaptive Client Selection in Resource Constrained Federated Learning Systems: A Deep Reinforcement Learning Approach," *IEEE Access*, vol. 9, pp. 98423-98432, 2021, doi: 10.1109/ACCESS.2021.3095915.
- [39] J. Li, Z. Yang, X. Wang, Y. Xia, and S. Ni, "Task offloading mechanism based on federated reinforcement learning in mobile edge computing," *Digital Communications and Networks*, vol. 9, no. 2, 2023, doi: 10.1016/j.dcan.2022.04.006.
- [40] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair Resource Allocation in Federated Learning," *arXiv*, 2020, doi: 10.48550/arXiv.1905.10497.

BIOGRAPHIES OF AUTHORS






Faustin Samafou    received B.Sc. degree in Electrical and Computer Engineering, with a specialization in Telecommunications, from Almergheb University, Alkhoms, Libya, in 2006. He received M.Sc. degree in Engineering Sciences, with a specialization in Computer Science, modelling and simulation of complex systems, from Polytechnic Graduate school, Cheikh Anta Diop University, research laboratory in Computer Science, Networks and Telecommunications, Dakar, Senegal, in 2016. He also received M.Sc. degree in Networks and Telecommunications, with a specialization in Radio communications and Services, from the Multinational Graduate School of Telecommunications in Dakar, Senegal, in 2017. He is working as a lecturer at National Higher School of Information and Communication Technology (ENASTIC). Currently, he is a Ph.D. student in Computer Science at the University of Maroua, Cameroon. His research interests include resource allocation and network slice orchestration in 5G and beyond, deep reinforcement learning, federated learning, and meta-learning. He can be contacted at email: faustin.samafou79@gmail.com.






Bakhit Amine Adoum    graduated from Universiti Teknologi PETRONAS (Malaysia) with Master and Ph.D. Degree in Electrical and Electronics Engineering field of Telecommunications, 2010 and 2014 respectively. His research interests include, but are not limited to wireless communication, cognitive radio, RF and microwave, cybersecurity, and machine learning. He published scientific articles on peer review journals, participated on international conferences and received several awards, including best paper award in NPC 2011, gold and silver medals at international invention and innovation fair (ITEX 2012, ACADREX 2012, ITEX 2013, SEDEX 29, Malaysia Technology Expo 2012, and Brussels Innova 2011). He held two Malaysian patents MY PI 2012000620 and MYPI 2013000128. He is a regular reviewer of international peer-reviewed journals. He currently holds the position of General Director at the National Higher School of Information and Communication Technology (ENASTIC) in Chad, as well as serving as an Assistant Professor at the National Higher Institute of Sciences and Techniques of Abeche (Chad), and Ambassador for ID4Africa. He was a research assistant at Universiti Teknologi PETRONAS, Malaysia. He can be contacted at email: bakhitammine@yahoo.fr.






Ado Adamou Abba Ari    is an Associate Researcher at the DAVID Lab of the Université Paris-Saclay, University of Versailles Saint-Quentin-en-Yvelines, France and Associate Professor at the LaRI Lab of the University of Maroua, Cameroon. He received his Ph.D. degree in Computer Science in 2016 from Université Paris-Saclay in France with the higher honors. He also received the Master Degree of Business Administration (MBA) in 2013, the Master of Science (M.Sc.) degree in Computer Engineering in 2012 and the Bachelor of Science (B.Sc.) degree in Mathematics and Computer Science in 2010 from the University of Ngaoundéré, Cameroon. He served/serving on several journals and conferences program and reviewing committees. Thus, he achieved the outstanding reviewer status with the Elsevier Computer Networks. Moreover he is recognized reviewer of a number of journals including Journal of Network and Computer Applications, IEEE Transactions on Intelligent Transportation Systems, IEEE Access, Information Sciences, Computer Communications, Telecommunication Systems, Journal of Ambient Intelligence and Humanized Computing. His current research is focused on wireless networks, security/privacy in IoT and artificial intelligence applications in communication networks. He can be contacted at email: adoadamou.abbaari@gmail.com.






Faïtchou Marius Fidel    was born on October 22, 1995 in Pala, Chad. He obtained his Bachelor's degree in Computer Science and Telecommunications from the University of Moundou, Chad, in 2017. He obtained his Master's degree in Telecommunications and Embedded Systems, Embedded Systems and Artificial Intelligence from the National School of ICT (ENASTIC), N'Djamena, Chad, in 2023. His master's thesis focuses on offloading and resource allocation in MEC-IoT based on federated meta-learning. He can be contacted at email: faitchoumarius@gmail.com.






Amir Mougache    is an Associate Professor at the University of N'Djamena. He is currently the Director of Academic Affairs and Examinations at the University of N'Djamena. He is also working a researcher with Laboratory of Information and Communication Technologies (LARTIC), Faculty of Exact and Applied Sciences of Farcha, University of N'djamena, Ndjamen, Chad. He can be contacted at email: amirmabate@gmail.com.



Nasrullah Armi    received Master Degree from Department of Knowledge based Information Engineering, Toyohashi University of Technology, Japan in 2004. Subsequently, he obtained his Ph.D. degree from Department of Electrical and Electronic Engineering, Universiti Teknologi Petronas, Malaysia in 2013. His research interests is signal processing, wireless communication and networks. He published scientific articles on peer review journals and participated on international conferences. He contributed his knowledge as scientific re-viewer in international impact journals and became Scientific Committee in International Conferences. Currently, he is working as a researcher with National Research and Innovation Agency, Research Center for Telecommunication, Bandung, Indonesia. He was an Assistant Professor in Jazan University, KSA from 2016-2018. He published scientific articles on peer review journals and conferences. He can be contacted at email: nasr004@brin.go.id.



Abdelhak Mourad Gueroui    is an Associate Professor in computer engineering at the University of Versailles Saint-Quentin-en-Yvelines, France. He received his M.Sc. degree from Université Paris 6 and his Ph.D. in networking and computer engineering in 2000 from the University of Versailles Saint-Quentin-en-Yvelines, France. He has authored more than 100 publications in international conferences and journals, as well as book chapters, including ACM, IEEE, ELSEVIER, and has both chaired and served in numerous program committees in prestigious international conferences. He received several awards, including best papers and serves/served on several journals and conferences executive committees. His research interests are in the area of wireless networks (WATM, WIMAX, LTE, 5G, cloud computing, WLAN, MESH, VANET, and WSNs), particularly performance evaluation and QoS provisioning. He can be contacted at email: mourad.gueroui@uvsq.fr.